DexMV: Imitation Learning for Dexterous Manipulation from Human Videos

Yuzhe Qin*, Yueh-Hua Wu*, Shaowei Liu, Hanwen Jiang, Ruihan Yang, Yang Fu, and Xiaolong Wang

University of California San Diego, La Jolla 92093, USA

A Overview

This supplementary material provides more details, results and visualizations accompanying the main paper. In summary, we include

– More details about video data collection;
– More details about the *Relocate*, *Pour*, and *Place inside* environments
– More details about demonstration translation;
– More visualization of hand-object pose estimation and hand motion retargeting results.

B Video Data Collection

We use Intel RealSense D435 cameras to collect human demonstrations of manipulating different objects for finishing diverse tasks on the table. In detail, each captured demonstration is about 10 seconds. Moreover, after the demonstration is captured, only the pour task is in need to reset the particles back into the mug, which tasks about 6 seconds. Thus, the time cost for data collection is not high, and the procedure tends to be scalable on different objects and tasks. In practice, it takes about 60 minutes to capture all 100 sequences for a task.

C Environments

We propose three types of manipulation tasks along with the DexMV Platform: *Relocate*, *Pour*, and *Place Inside*. The manipulated objects come from YCB Dataset [1]. Environments use the MuJoCo simulator [6] with timestep set to 0.002 and frame skip set to 5. We adopt the same contact friction parameters following the setting in the literature [5]. We use the open-source MuJoCo model of Adroit Hand [1]. Figure 1 shows the seven different task used in DexMV:*Relocate* with five different objects, *Pour*, and *Place Inside*.

Action. The action space is the same for all tasks, which is the motor command of 30 actuators on the robotic hand. The first 6 motors control the global position and orientation of the robot while the last 24 motors control the fingers of the hand. We normalize the action range to $(-1, 1)$ based on actuator specification.

[1] https://github.com/vikashplus/Adroit

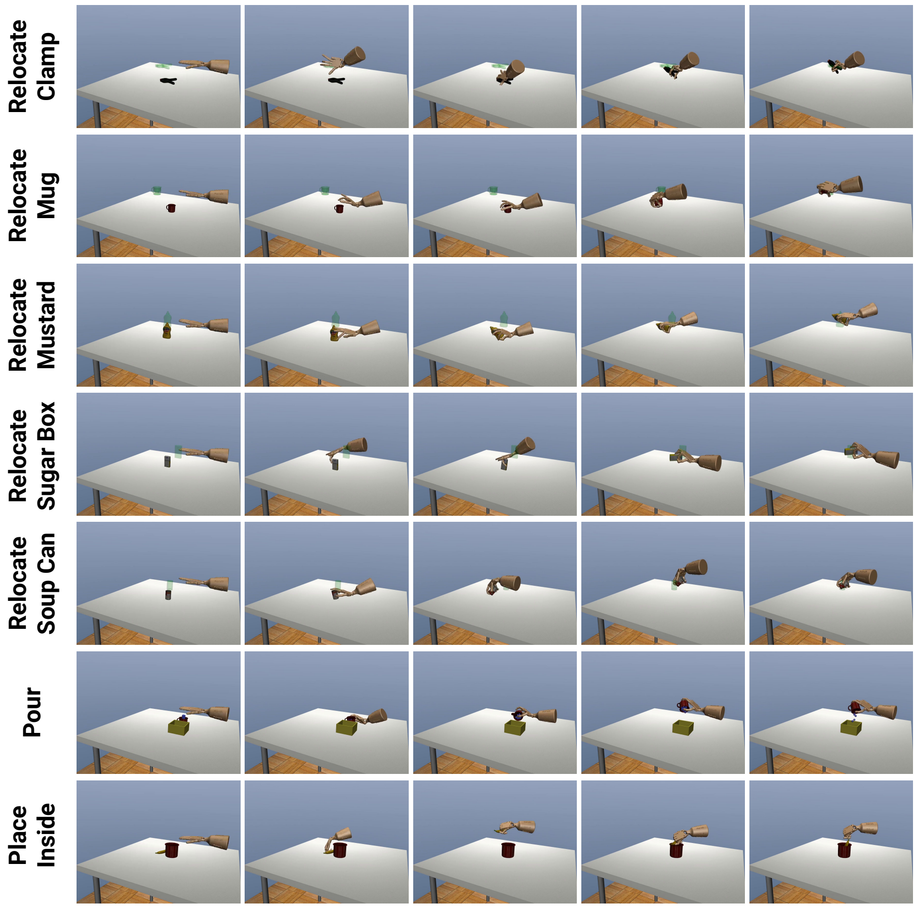

Fig. 1: **DexMV Tasks.** There are three types of tasks in the figure. The first five columns: *relocate* with mug, mustard bottle, clamp, sugar box, tomato soup can. *Relocate* task means that the agent needs to move the object from the initial position to the target position. The sixth and seventh columns: *Pour* and *Place Inside*. In *Pour* task, the agent needs to pour the water particles inside a mug into the yellow container. In *Place Inside* task, the agent needs to manipulate the orientation of the banana to place it inside a mug. Both of the last two tasks require delicate manipulation.

C.1 Relocate

Observation. The observation of *Relocate* is composed of four components: (i) joint angles of adroit robotic hand; (ii) global position of adroit hands root; (iii) object position; (iv) target position. The overall observation space is 39-dim.

Reward. The reward is defined based on three distances: (i) the distance between the robot hand and the object; (ii) the distance between the robot hand and the target; (iii) the distance between the object and the target. Lower distance corresponds to higher reward.

Reset. For each episode, the xy position of both object and target is randomized within a $(-0.3, 0.3)$ square on the table. The height of target is randomized between $(0.15, 0.25)$.

C.2 Pour

Observation. Similar to *Relocate*, we include the robot joint angles, root position of robot hand, and object position in the observation. In *Pour*, we replace the target position with the container position. Besides, since the agent needs orientation information of the mug to pour the water particles, we add a quaternion to represent object orientation.

Reward. The main reward is based on the final number of particles that fell within the container. Additionally, similar to *Relocate*, we use the distance between the robot hand and the object as well as the distance between the object and the container to provide part of the rewards. Lower distance leads to higher reward. The coefficient of the main reward is $10\times$ larger than the reward computed based on the distance.

Reset. For each episode, the xy position of the mug is randomized between $(-0.1, 0.1)$ on the table. The water particles are inside the mug at the beginning of each episode. The container is always at the center of the tabletop.

C.3 Place Inside

Observation. The observation space of the *Place Inside* task is the same as the observation space of *Pour* as described in Section C.2.

Reward. The main reward is based on the position and orientation of the manipulated object. If the object is placed inside of the mug, the agent will get a large portion of the reward. Similar to *Relocate*, we also add a lifting reward to encourage the robot to first lift the object before moving it towards the container.

Reset. For each episode, the xy position of the object is randomized between $(-0.15, 0.15)$ on the table. The container, i.e. mug, is always placed at the center of tabletop.

C.4 ShapeNet Objects

In the *Generalization on Novel Objects and Category* experiments (Section 8.4) of the main paper, we use objects from ShapeNet [2] dataset. We pre-process the ShapeNet geometry by scaling the mesh so that it can be used in our simulated environment. The YCB objects [1] are captured from the real scan, so the scale of object mesh from YCB dataset aligns with the real counterpart and can be used for robot manipulation directly. Different from the YCB dataset, the ShapeNet dataset does not contain any scale information, e.g. the mug in ShapeNet can be larger than the robot. So we need to scale the object to a reasonable size in which it can be manipulated by the robot. We scale the object based on the diagonal length of the object bounding box. For each category, we manually select a diagonal length so that all object instances from the category will have the same bounding-box diagonal length after scaling. Besides, we will not use objects with non-manifold geometry to avoid instability in physical simulation. For the *Place Inside* task, the object mesh should be watertight for volume computation. We use the convex meshes processed by VHACD [4] for both simulation and volume computation.

D Demonstration Translation

D.1 Kinematics Model

In Table 1, we compare the difference of kinematics model between the human hand and robot hand. The overall Degree-of-Freedom(DoF) of the human hand is higher than the DoF of the robot hand. Thus hand motion retargeting from human to robot is projecting a pose from a higher dimension to a lower dimension, which will lose information inevitably. The motion retargeting module try to maintain the task space vectors between these two different kinematics model.

	H-Joints	**H**-DoF	**R**-Joints	**R**-DoF
Thumb	3x Ball	9	5x Revolute	5
Index	3x Ball	9	4x Revolute	4
Middle	3x Ball	9	4x Revolute	4
Ring	3x Ball	9	4x Revolute	4
Pinkie	3x Ball	9	5x Revolute	5
Wrist	Null	0	2x Revolute	2
Root	1x Free	6	1x Free	6
Overall	N/A	51	N/A	30

Table 1: **Comparison of Kinematics** We compare the kinematic model between MANO human hand and the robot hand we used in simulator. **H** is the abbr for Human while **R** stands for robot. For example, the **H**-Joints column shows the number and type of joints for a specific sub-part in the kinematics model. Each ball joint has 3 Degree-of-Freedom(DoF), each revolute joint has 1 DoF, and each free joint has 6 DoF

D.2 Initialization in Hand Motion Retargeting

As mentioned in the Demonstration Translation section of the main paper, the robot joint angles are solved using optimization. A good initialization is essential for optimization. For $t \geq 1$, q_t is initialized using the optimization results

q_{t-1}. Here we will discuss how to initialize q_t for $t = 0$. Previous work [3] initializes q_t with zero vectors. The optimization cannot provide reasonable outputs when the goal is far from zeros. To tackle this issue, we use a heuristic function $\phi(\theta_0)$ to initialize q_0. The heuristic function takes the MANO hand pose parameters θ_0 and outputs an estimation of the robot joint angles. The heuristic function $\phi(\theta_0)$ can be regarded as a coarse hand motion retargeting function which does not consider the shape difference between human and robot hands. It is only used to provide a reasonable initialization for further optimization.

Given the axis-angle representation θ_0 from hand pose estimation, we first convert it to 15 rotation matrices. Then we compute the projection of each rotation matrix on the joint axis direction of each robot hand joint. Then we use a manually-designed vector w to map the projection to the robot joint angle. The overall function can be formulated as below,

$$\phi(\theta_0) = \mathbf{P_{rot}}(\mathbf{R}(\theta_0))w \tag{1}$$

where $\mathbf{R}(\cdot)$ is the Rodrigues' formula that maps axis-angle to rotation matrix, $\mathbf{P_{rot}}(\cdot)$ is the projection function to compute the nearest rotation along with the robot joint direction for each joint, i.e. projection. $\mathbf{P_{rot}}(\mathbf{R}(\theta_0)) \in \mathbb{R}^{24\times 15}$ and $w \in \mathbb{R}^{15}$ is a manually designed vector. Thus the dimension of $\phi(\theta_0)$ is 24, which corresponds to the joint angles for finger and wrist. As mentioned in Table 1, the overall DoF of the robot hand is $24 + 6 = 30$, which includes 6 DoF hand root pose. We directly use the root position plus root orientation from human hand pose estimation as the root pose for the robot hand.

D.3 Post Processing

Filter Estimated Pose: When human is manipulating the object, either hand or object is in heavy occlusion, which may cause inconsistent estimation results. To improve the temporal consistency of the estimated hand and object poses, we apply a digital low-pass filter to remove the high-frequency noise. The sampling frequency of the filter is 100 while the cutoff frequency is 5 for the position of both object and hand. Filtering the rotation is not as straightforward as filtering the position. To get a smooth orientation sequence, we first convert the rotation into $so(3)$ lie algebra. Then, we apply the filter in $so(3)$ space and convert it back to rotation matrix $SO(3)$ after filtering.

Hindsight Goal Position for Relocate. As mentioned in the Task Section of the main paper, *relocate* is a goal-conditioned task. The goal information should also be included in the state representation. To provide goal information from human demonstration, we use the position of the object in the last step as the hindsight goal.

Frame Alignment. Spatial quantities like object pose are dependent on the frame in which it is observed. The natural frame for pose estimation results is the camera frame. In the simulated environment, such a camera frame does not exist and the pose is represented in the world coordinate fixed on the table. Thus we also align the frame in demonstrations to match the simulated environment.

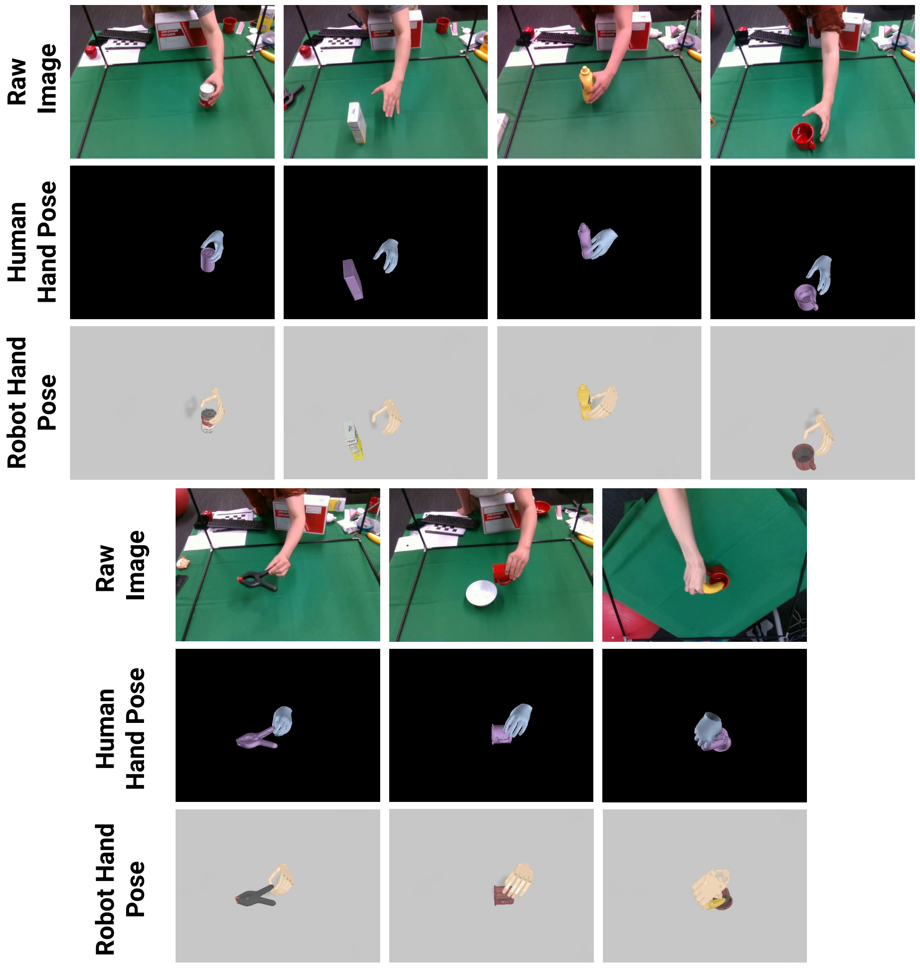

Fig. 2: **3D hand-object pose estimation results and hand motion retargeting results.** Visualization on relocate tomato soup can, sugar box, mustard bottle, mug, clamp, pour, and place inside.

E More Visualization of Hand-Object Pose Estimation and Hand Motion Retargeting

In this section, we provide more visualization on hand-object pose estimation and hand motion retargeting in Figure 2. The four tasks in the first three rows are *Relocate* with tomato soup can, sugar box, a mustard bottle, and mug. The three tasks in the last three rows are: *Relocate* with clamp, *Pour*, and *Place Inside*.

References

1. Calli, B., Walsman, A., Singh, A., Srinivasa, S., Abbeel, P., Dollar, A.M.: Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols. arXiv (2015) 1, 3
2. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015) 3
3. Handa, A., Van Wyk, K., Yang, W., Liang, J., Chao, Y.W., Wan, Q., Birchfield, S., Ratliff, N., Fox, D.: Dexpilot: Vision-based teleoperation of dexterous robotic hand-arm system. In: ICRA (2020) 5
4. Mamou, K., Ghorbel, F.: A simple and efficient approach for 3d mesh approximate convex decomposition. In: 2009 16th IEEE international conference on image processing (ICIP). pp. 3501–3504. IEEE (2009) 4
5. Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., Levine, S.: Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. arXiv (2017) 1
6. Todorov, E., Erez, T., Tassa, Y.: Mujoco: A physics engine for model-based control. In: IROS (2012) 1